

DOKUMENTATION-FRIDGE INSIGHT

I. Zielbestimmung

Fridge-Insight ist eine Software, mit der Sie Ihren Kühlschrank besser verwalten können. Es enthält viele gut programmierte Funktionen, um ein optimales Verwaltungssystem zu gewährleisten.

II. Muss-Kriterien

Das System muss die Möglichkeiten bieten:

- ✓ Benutzer zu verwalten.
 - ❖ Konto addieren
 - ❖ Konto bearbeiten
 - ❖ Konto entfernen
- ✓ Benachrichtigungen zu konfigurieren
 - ❖ Benachrichtigungen erstellen
 - ❖ Benachrichtigungen bearbeiten
 - ❖ Benachrichtigungen entfernen
- ✓ Einkauf Liste zu verwalten
 - ❖ Einkaufskorb erstellen
 - ❖ Element zum Einkaufskorb hinzufügen
 - ❖ Element vom Einkaufskorb entfernen
- ✓ Hinzufügen/Entfernen eines Elements
 - ❖ Produkt addieren
 - ❖ Produkt bearbeiten
 - ❖ Produkt entfernen

III. Wunsch-Kriterien

- ✓ Die Menge der Lebensmittel sollte automatisch aktualisiert werden
- ✓ Die Benachrichtigungen sollen bei Lebensmittelknappheit automatisch erstellt werden
- ✓ Jeder Benutzer hat die Möglichkeit, ein Profil zu erstellen, in dem er Lieblingsessen und andere Funktionen ablegen kann.

IV. Abgrenzungs-Kriterien

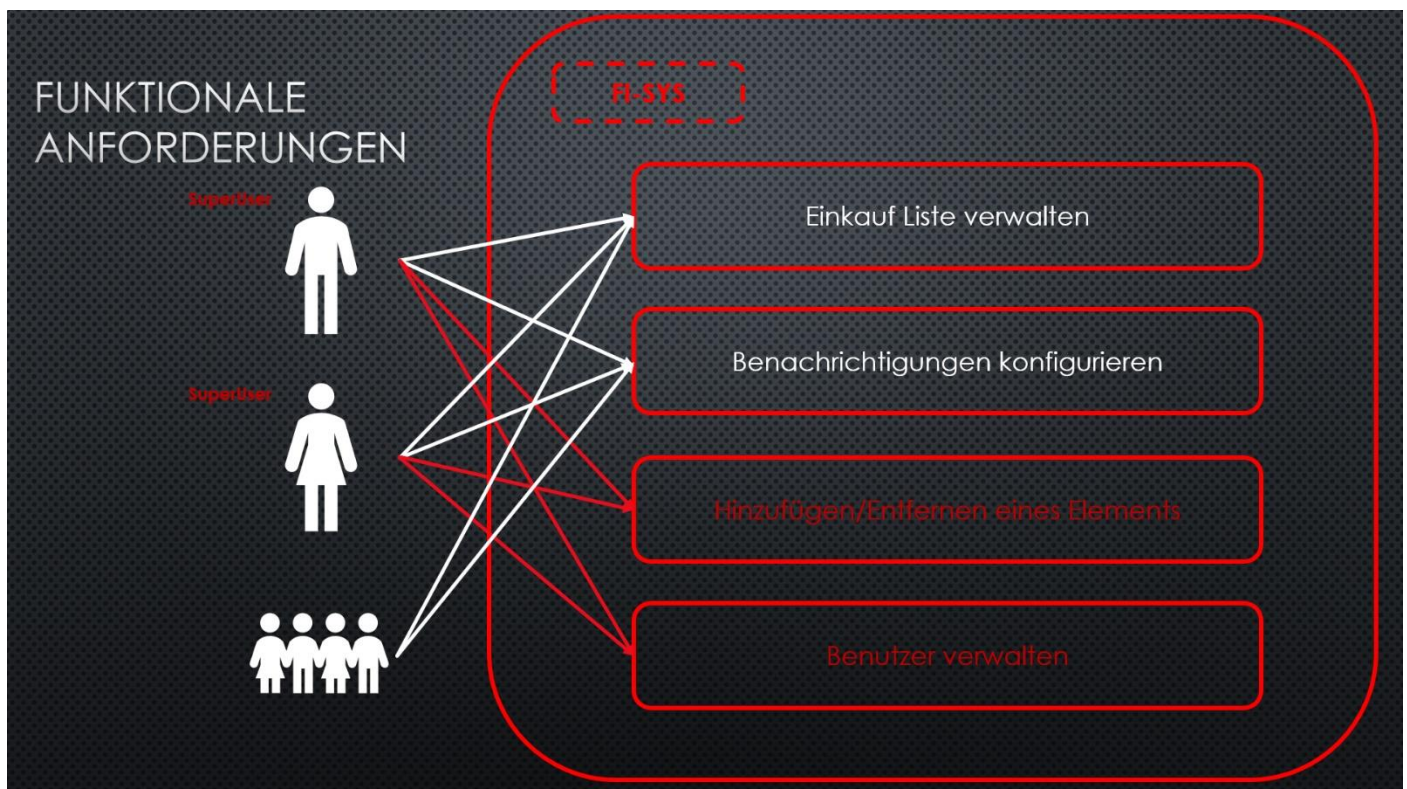
- ✓ Nur der "SuperUser" hat die Berechtigung, ein Konto hinzuzufügen, zu ändern oder zu entfernen.
- ✓ Nur ein "SuperUser" kann ein Produkt hinzufügen, ändern oder löschen
- ✓ Nur ein "SuperUser" kann ein anderes Konto als "SuperUser" kennzeichnen.

V. Akteure: Berechtigungen und Begrenzungen

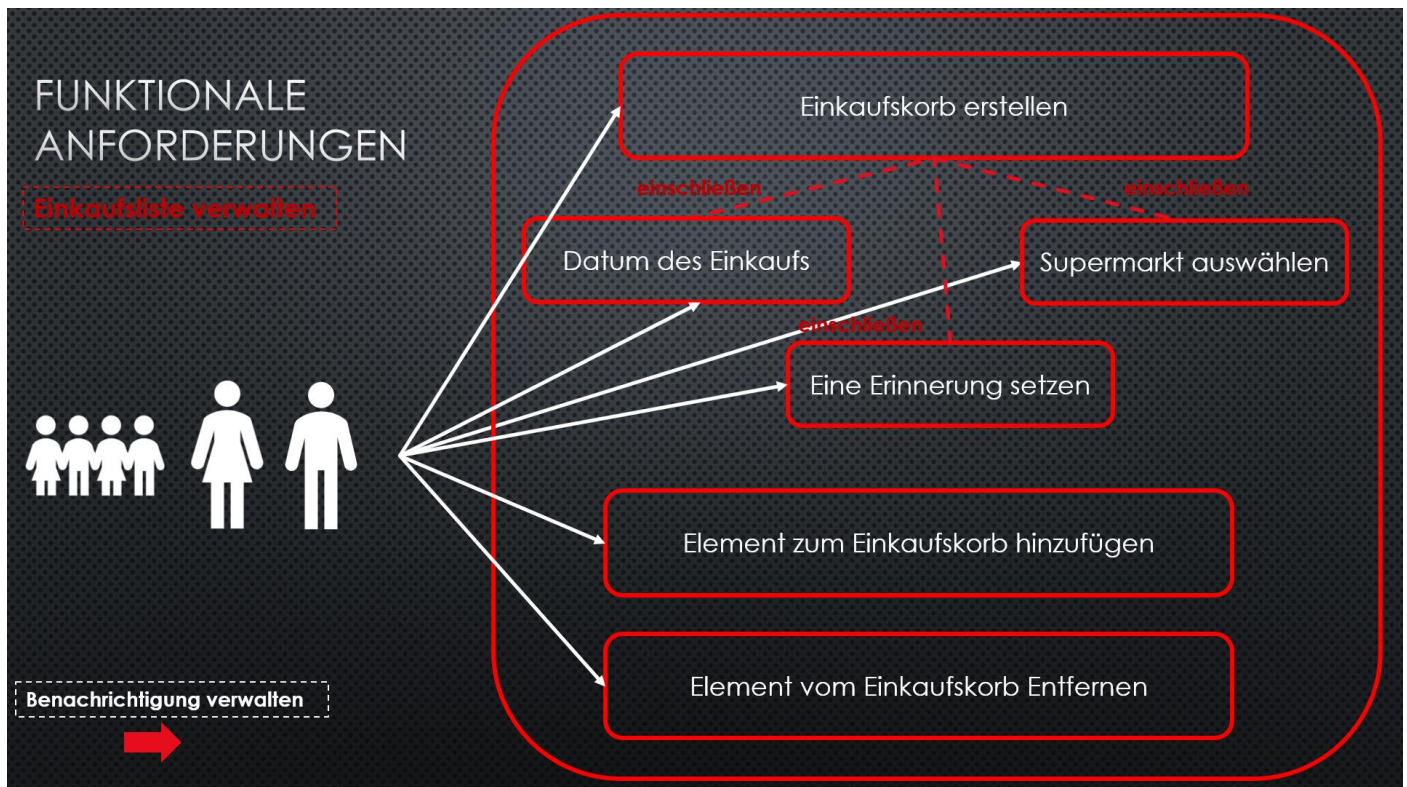
Rolle	Beschreibung	Vertreter(Bsp.)	Bemerkungen
Super User	Der Superuser kann auf alle Funktionen des Programms zugreifen.	Vater/Mütter	
User	Der User kann auf alle grundlegenden Funktionen zugreifen	Kinder/Freunde	Vertreter kann Super User werden, aber er braucht die Erlaubnis eines der Super Users

VI. Funktionale Anforderungen

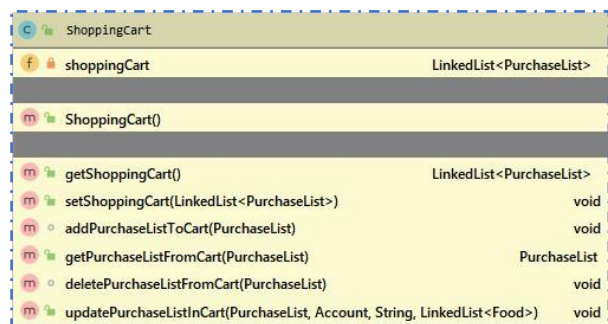
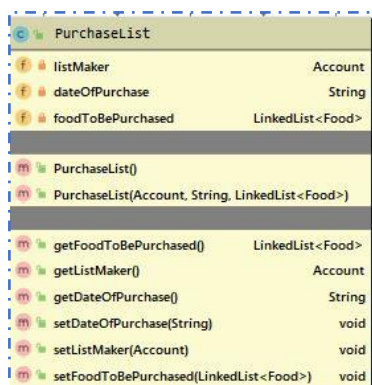
1. Grobe Übersicht



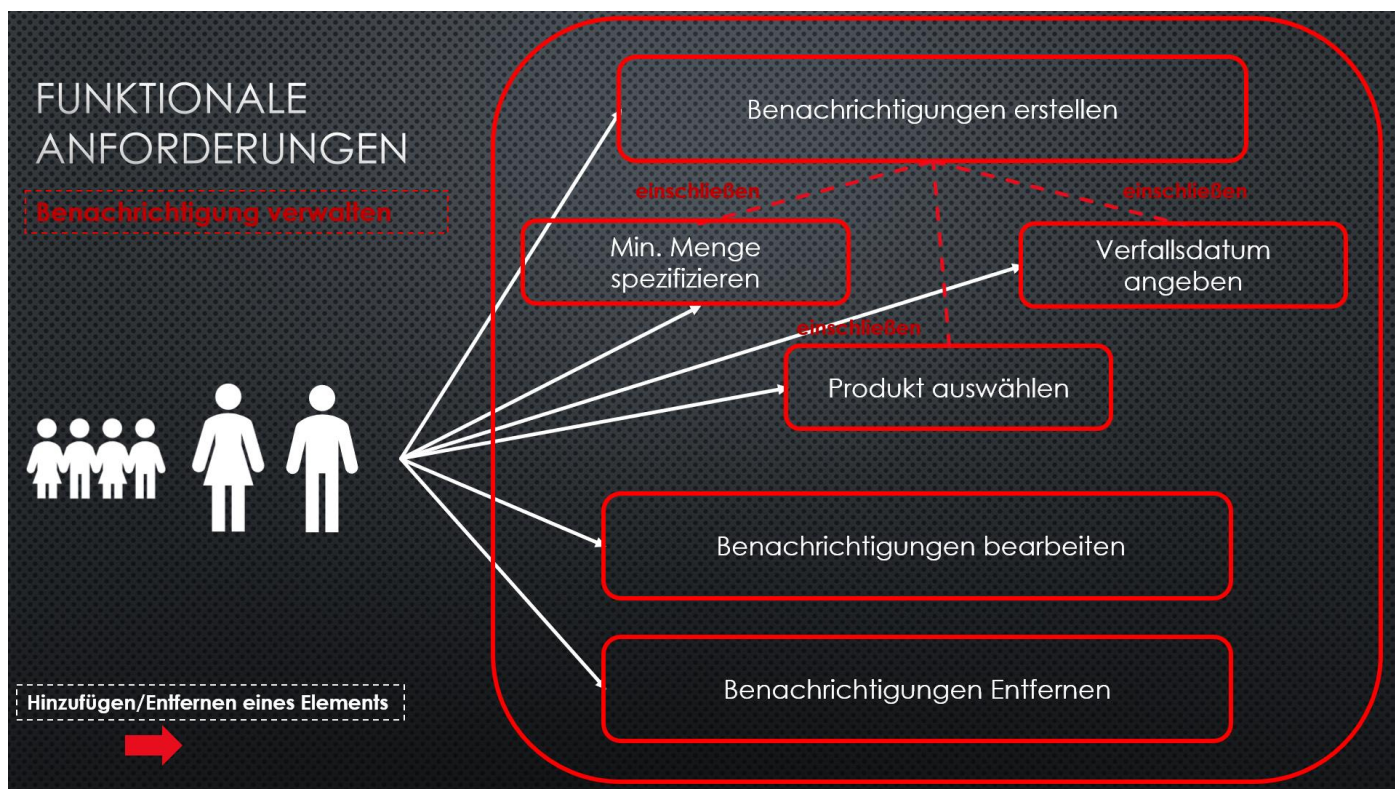
2. Anwendungsfälle
 - a. Einkauf Liste verwalten



✓ Klassen, die in diesem Fall verwendet werden, sind



b. Benachrichtigungen konfigurieren

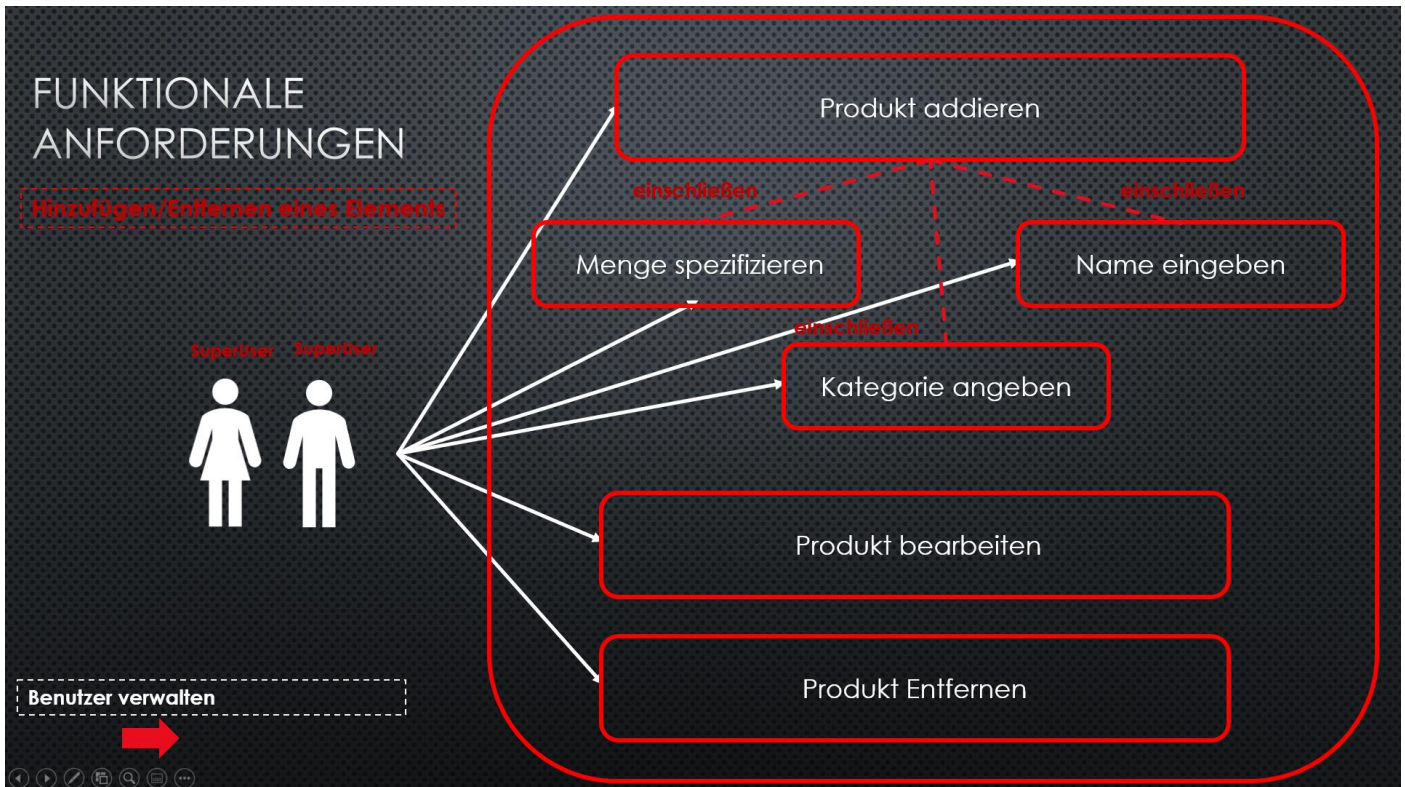


✓ Klassen, die in diesem Fall verwendet werden, sind

NotificationFood	
minimumQuantity	int
selectedFood	Food
dateOfExpiration	String
NotificationFood()	
NotificationFood(int, Food, String)	
getSelectedFood()	Food
getMinimumQuantity()	int
setMinimumQuantity(int)	void
setSelectedFood(Food)	void
getDateOfExpiration()	String
setDateOfExpiration(String)	void

NotificationFoodList	
Notifications	LinkedList<NotificationFood>
NotificationFoodList()	
getNotifications()	LinkedList<NotificationFood>
setNotifications(LinkedList<NotificationFood>)	void
addNotificationToList(NotificationFood)	void
deleteNotificationFromList(NotificationFood)	void
updateNotificationInList(NotificationFood, String, int, Food)	void
getNotificationFromList(NotificationFood)	NotificationFood

c. Hinzufügen/Entfernen eines Elements



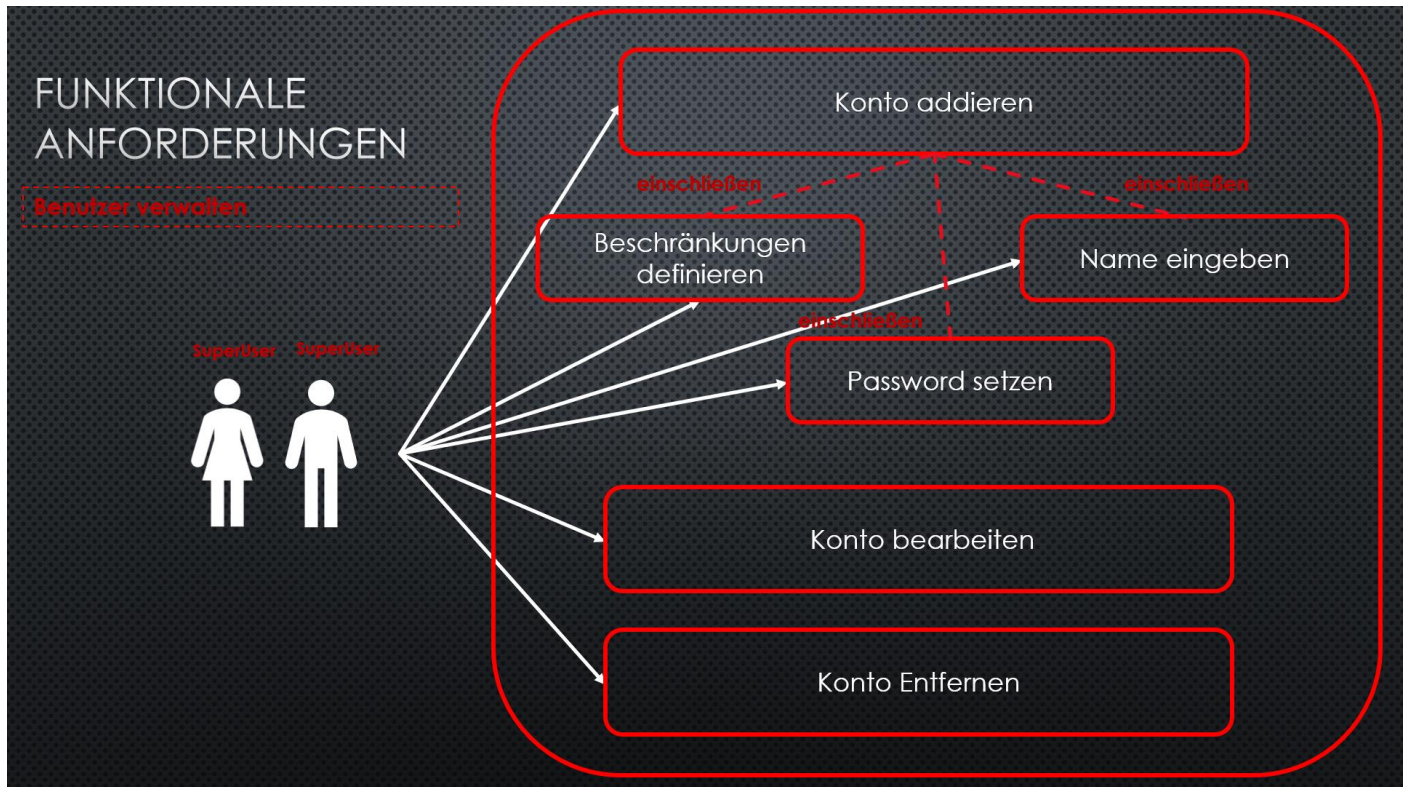
✓ Klassen, die in diesem Fall verwendet werden, sind

Address	
houseNumber	int
streetName	String
City	String
State	String
postalCode	int
Address()	
Address(int, String, String, String, int)	
getHouseNumber()	int
getPostalCode()	int
getCity()	String
getState()	String
getStreetName()	String
setCity(String)	void
setHouseNumber(int)	void
setPostalCode(int)	void
setState(String)	void
setStreetName(String)	void

SuperMarket	
marketName	String
marketAddress	Address
SuperMarket()	
SuperMarket(String, Address)	
getMarketAddress()	Address
getMarketName()	String
setMarketAddress(Address)	void
setMarketName(String)	void

Food	
foodTitle	String
expireDate	String
foodType	String
foodQuantity	int
foodMarket	SuperMarket
Food()	
Food(String, String, String, int, SuperMarket)	
getFoodTitle()	String
getFoodType()	String
getExpireDate()	String
getFoodQuantity()	int
getFoodMarket()	SuperMarket
setExpireDate(String)	void
setFoodMarket(SuperMarket)	void
setFoodQuantity(int)	void
setFoodTitle(String)	void
setFoodType(String)	void

d. Benutzer verwalten

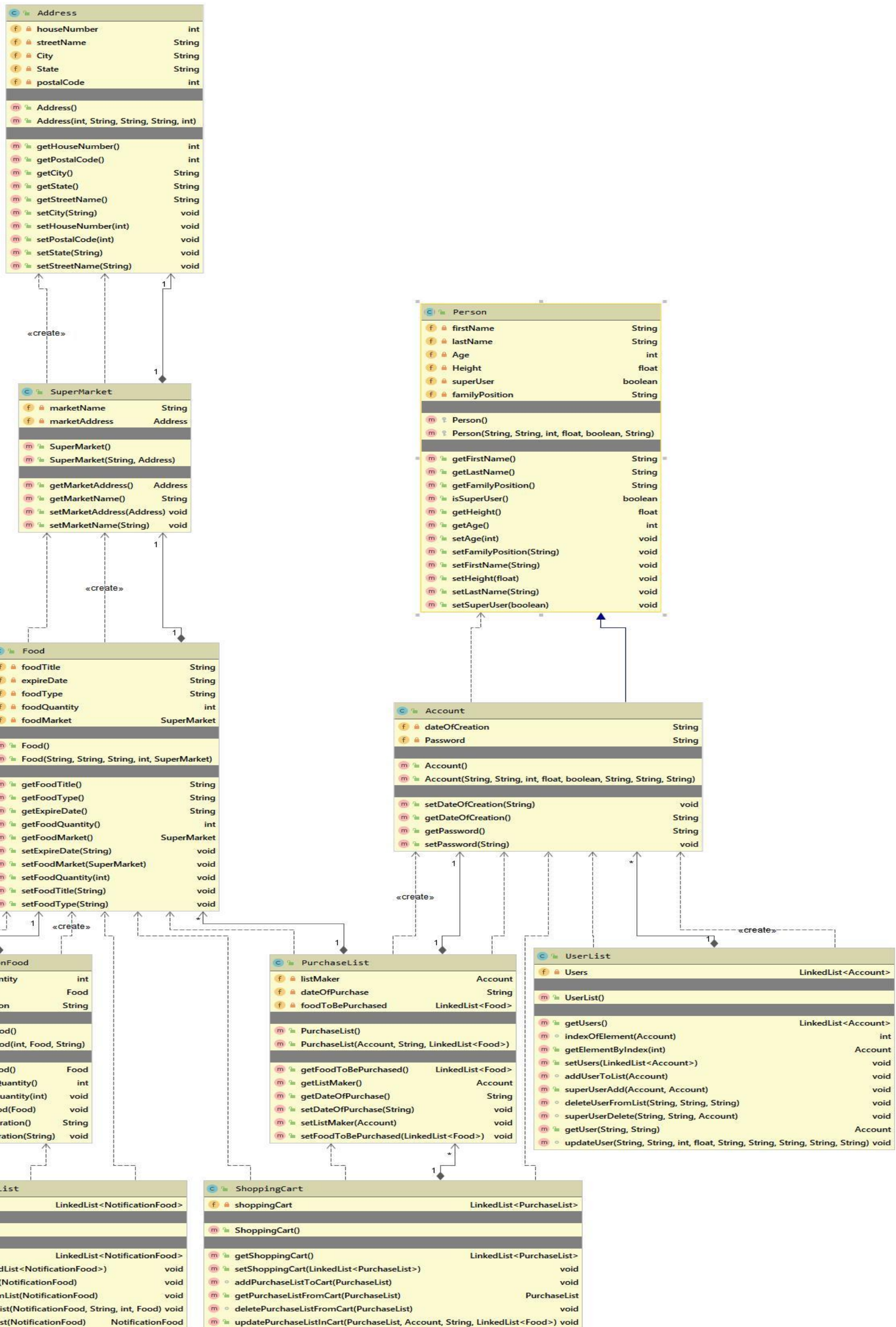


✓ Klassen, die in diesem Fall verwendet werden, sind

Person	
f	firstName String
f	lastName String
f	Age int
f	Height float
f	superUser boolean
f	familyPosition String
m	Person()
m	Person(String, String, int, float, boolean, String)
m	getFirstName() String
m	getLastName() String
m	getFamilyPosition() String
m	isSuperUser() boolean
m	getHeight() float
m	getAge() int
m	setAge(int) void
m	setFamilyPosition(String) void
m	setFirstName(String) void
m	setHeight(float) void
m	setLastName(String) void
m	setSuperUser(boolean) void

Account	
f	dateOfCreation String
f	Password String
m	Account()
m	Account(String, String, int, float, boolean, String, String, String)
m	setDateOfCreation(String) void
m	getDateOfCreation() String
m	getPassword() String
m	setPassword(String) void

UserList	
f	Users LinkedList<Account>
m	UserList()
m	getUsers() LinkedList<Account>
m	indexOfElement(Account) int
m	getElementByIndex(int) Account
m	setUsers(LinkedList<Account>) void
m	addUserToList(Account) void
m	superUserAdd(Account, LinkedList<Account>, Account) LinkedList<Account>
m	deleteUserFromList(String, String, String) void
m	superUserDelete(String, String, Account) void
m	getUser(String, String) Account
m	updateUser(String, String, int, float, String, String, String, String, String) void



VIII. UML-Diagramm mit Test Klassen (die Beziehungen zwischen die unterschiedlichen Klassen des Projekts)

